Noname manuscript No.
(will be inserted by the editor)

Analysis of I/O Performance on an Amazon EC2 Cluster Compute and High I/O Platform

Roberto R. Expósito · Guillermo L. Taboada · Sabela Ramos · Jorge González-Domínguez · Juan Touriño · Ramón Doallo

Abstract Cloud computing is currently being explored by the scientific community to assess its suitability for High Performance Computing (HPC) environments. In this novel paradigm, compute and storage resources, as well as applications, can be dynamically provisioned on a pay-per-use basis. This paper presents a thorough evaluation of the I/O storage subsystem using the Amazon EC2 Cluster Compute platform and the recent High I/O instance type, to determine its suitability for I/O-intensive applications. The evaluation has been carried out at different layers using representative benchmarks in order to evaluate the low-level cloud storage devices available in Amazon EC2, ephemeral disks and Elastic Block Store (EBS) volumes, both on local and distributed file systems. In addition, several I/O interfaces (POSIX, MPI-IO and HDF5) commonly used by scientific workloads have also been assessed. Furthermore, the scalability of a representative parallel I/O code has also been analyzed at the application level, taking into account both performance and cost metrics. The analysis of the experimental results has shown that available cloud storage devices can have different performance characteristics and usage constraints. Our comprehensive evaluation can help scientists to increase significantly (up to several times) the performance of I/O-intensive applications in Amazon EC2 cloud. An example of optimal configuration that can maximize I/O performance in this cloud is the use of a RAID 0 of 2 ephemeral disks, TCP with 9,000 bytes MTU, NFS async and MPI-IO on the High I/O instance type, which provides ephemeral disks backed by Solid State Drive (SSD) technology.

Keywords Cloud Computing · Virtualization · I/O Performance Evaluation · Network File System (NFS) · MPI-IO · Solid State Drive (SSD)

Roberto R. Expósito · Guillermo L. Taboada · Sabela Ramos · Jorge González-Domínguez · Juan Touriño · Ramón Doallo
Computer Architecture Group, Dept. of Electronics and Systems,
University of A Coruña (Spain)
E-mail: {rreye,taboada,sramos,jgonzalezd,juan,doallo}@udc.es

1 Introduction

Data management is a critical component of many current scientific computing workloads, which are generating very large data sets, contributing significantly to the consolidation of the so-called current *big data* era. These applications often require a high number of computing resources to perform large-scale experiments into a reasonable time frame, and these needs have been typically addressed with dedicated High Performance Computing (HPC) infrastructures such as clusters or big supercomputers. In this scenario, scientific applications can be sensitive to CPU power, memory bandwidth/capacity, network bandwidth/latency as well as the performance of the I/O storage subsystem.

The cloud computing paradigm is a relatively recent computing model where dynamically scalable and often virtualized resources are provided as a service over the Internet. This novel paradigm has gained significant popularity in many areas, including the scientific community. The combination of this model together with the rich set of cloud infrastructure services can offer a feasible alternative to traditional servers and computing clusters, saving clients from the expense of building an in-house datacenter that is provisioned to support the highest predicted load. With cloud-based technologies, scientists can have easy access to large distributed infrastructures and completely customize their execution environment, thus providing the perfect setup for their experiments. In addition, the interest in the use of public clouds for HPC applications increases as their availability, computational power, price and performance improves.

Amazon Web Services (AWS) is nowadays the leading public Infrastructure-as-a-Service (IaaS) cloud provider in terms of number of users, allowing resources in their data centers to be rented on-demand through Elastic Compute Cloud (EC2) service [7]. By means of virtualization technologies, EC2 allows scalable deployment of applications by providing a web service through which a user can, among other tasks, boot straightforwardly an Amazon Machine Image (AMI) into a custom Virtual Machine (a VM or "instance"). This on-demand allocation of resources provides a new dimension for HPC due to the elastic capability of the cloud computing model, which additionally can provide both cost-effective and energy-efficient solutions [31].

Amazon EC2 offers a cloud infrastructure, the Cluster Compute (CC) platform, which specifically targets HPC environments [9]. The CC platform is a family of several instance types which are intended to be well suited for large-scale scientific experiments and HPC applications by offering physical node allocation (a single VM per node), powerful and up-to-date CPUs and GPUs, and an improved interconnection network (10 Gigabit Ethernet). Additionally, the High I/O instance type shares the same characteristics as the CC instances with enhanced storage performance providing Solid State Drives (SSD) disks. Using these instance types customers can expedite their HPC workloads on elastic resources as needed, adding and removing compute resources to meet the size and time requirements for their specific workloads. An example of the extent and magnitude of Amazon EC2 is the self-made cluster that, with only a small portion of its resources (about 1,000 CC instances), ranks #102 in the latest Top 500 list (November 2012) [1].

This paper evaluates the I/O storage subsystem on the Amazon EC2 CC platform to determine its suitability for scientific applications with high I/O performance requirements. Moreover, the evaluation includes, for the first time to the best of our knowledge, the High I/O instance type, which has been recently released in July 2012. This instance type is intended to provide very high instance storage I/O performance, as it is backed by SSD disks, which is the main differential characteristic of this resource, but it also provides high levels of CPU, memory and network performance as CC instances.

In this evaluation, experiments at different levels are conducted. Thus, several micro-benchmarks are used to evaluate different cloud low-level storage devices available in CC instances, ephemeral disks and Elastic Block Store (EBS) volumes [6], both at the local and distributed file system levels. In addition, common middleware libraries such as HDF5 [4] and MPI-IO [35], which are directly implemented on top of file systems, have also been assessed as scientific workloads usually rely on them to perform I/O. Finally, the scalability of a representative parallel I/O code implemented on top of MPI-IO, the BT-IO kernel [38] from the NAS Parallel Benchmarks (NPB) suite [24], has also been analyzed at the application level both in terms of performance and cost effectiveness.

The paper is organized as follows: Section 2 describes the related work. Section 3 presents an overview of the storage system of the Amazon EC2 public cloud. Section 4 introduces the experimental configuration, both hardware and software, and the methodology of the evaluation conducted in this work. Section 5 analyzes the I/O performance results of the selected benchmarks/kernels on Amazon EC2 CC and High I/O instances. Finally, our conclusions are presented in Section 6.

2 Related Work

In recent years there has been a spur of research activity in assessing the performance of virtualized resources and cloud computing environments [18,25,30,40,41]. The majority of recent studies have evaluated Amazon EC2 to examine the feasibility of using public clouds for high performance or scientific computing, but with different focuses.

Some previous works have shown that computationally-intensive codes present little overhead when running on virtualized environments, whereas communication-intensive applications tend to perform poorly [12,13,23,27,37], especially tightly-coupled parallel applications such as MPI [3] jobs. This is primarily due to the poor virtualized network performance, processor sharing among multiple users and the use of commodity interconnection technologies (Gigabit Ethernet). In order to overcome this performance bottleneck, Amazon EC2 offers the Cluster Compute (CC) platform, which introduces several HPC instance types, cc1.4xlarge and cc2.8xlarge, abbreviated as CC1 and CC2, respectively, in addition to the recent High I/O instance type (hi1.4xlarge, abbreviated as HI1) which provides SSD disks. Thus, Sun et al. [34] relied on 16 CC1 instances for running the Lattice Optimization and HPL benchmark. The main conclusion derived from the results is that MPI codes, especially those which are network latency bound, continue to present poor scalability. Ramakrishnan et al. [29] stated that virtualized network is the main performance bottleneck on Amazon EC2 after analyzing the communication overhead on CC1 instances. Mauch et al. [21] presented an overview of the current state of HPC IaaS offerings and suggested how to use InfiniBand in a private virtualized environment, showing some HPL benchmark results using a single instance of CC1 and CC2 instance types. Finally, our previous work [14] has stated that CC1 and CC2 instances are able to achieve reasonable scalable performance in parallel applications, especially when hybrid shared/distributed memory programming paradigms, such as MPI+OpenMP, are used in order to minimize network communications.

However, most of the previous work is focused on computation and communication, whereas there are very little works that have investigated I/O and storage performance. Some of them analyzed the suitability of running scientific workflows in the cloud [19,26,36], showing that it can be a successful option as these workloads are loosely-coupled parallel applications. Thus, Juve et al. [19] studied the performance and cost of different storage options for scientific workflows on Amazon EC2, although regarding CC platform they only evaluated three workflows on the CC1 instance type. Vecchiola et al. [36] ran an fMRI brain imaging workflow on Amazon EC2 using the object-based Amazon Simple Storage Service (S3) [8] for storage, and analyzed the cost varying the number of nodes. In [5], Abe and Gibson provide S3-like storage access on top of PVFS [10] on an open-source cloud infrastructure. Palankar et al. [28] assessed the feasibility of using Amazon S3 for scientific grid computing. Zhai et al. [42] conducted a comprehensive evaluation of MPI applications on Amazon EC2 CC platform, revealing a significant performance increase compared to previous evaluations on non-CC instances. They also reported some experimental results of storage performance, but limited to ephemeral devices (local disks) without RAID and using only CC1 instances. In [32], the storage and network performance of the Eucalyptus cloud computing framework is analyzed, confronted with some results from one large instance type of Amazon EC2. Ghoshal et al. [16] compared I/O performance on two cloud platforms, Amazon EC2 and Magellan, using IOR benchmark on the CC1 instance type. Their study is limited to the file system level, so RAID configurations as well as the performance of I/O interfaces are not taken into account. Finally, Liu et al. [20] ran two parallel applications (BT-IO and POP) on CC1 instances using ephemeral disks, both for NFS and PVFS file systems. Their results show that cloud-based clusters enable users to build per-application parallel file systems, as a single parallel I/O solution can not satisfy the needs of all applications.

In addition, many current applications (e.g., data mining, social network analysis) demand distributed computing frameworks such as MapReduce [11] and iMapReduce [43] to process massive data sets. An attractive feature of these frameworks is that they support the analysis of petabytes of data with the help of cloud computing without any prior investment in infrastructure, which has popularized big data analysis. For instance, Gunarathne et al. [17] present a new MapReduce runtime for scientific applications built using the Microsoft Azure cloud infrastructure. In [39], Yang et al. proposed a regression model for predicting relative performance of workloads under different Hadoop configurations with 87% accuracy.

In this paper we evaluate the I/O performance of an Amazon EC2 CC and High I/O platform. Thus, we evaluated CC1 and CC2 instance types together with the most recent HI1 instances, so they can be directly compared. Moreover, we analyze the performance of the different low-level storage devices available on these instances (EBS volumes and ephemeral disks) in addition to the use of software RAID. Moreover, our study is carried out at several layers (storage devices, file systems, I/O interfaces and applications) using representative benchmarks/applications for each layer. Finally, we also take into account the costs associated with the use of a public cloud infrastructure, presenting a cost analysis at the application level.

3 Overview of Amazon EC2 CC and High I/O Instances

Amazon EC2 offers the CC platform which currently provides two HPC instance types. The Cluster Compute Quadruple Extra Large instances (cc1.4xlarge, abbreviated as CC1) and Cluster Compute Eight Extra Large instances (cc2.8xlarge, abbreviated as CC2) are resources with 23 and 60.5 GBytes of memory and 33.5 and 88 EC2 Compute Units (ECUs) for CC1 and CC2, respectively. According to Amazon WS one ECU provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.

In addition to the CC instances, Amazon has recently launched (July 2012) the High I/O Quadruple Extra Large instances (hi1.4xlarge, abbreviated as HI1). These instances have two SSD disks as local block storage, which is the main differential characteristic of this resource, in order to provide very high instance storage I/O performance. Moreover, HI1 instances also have powerful CPUs (35 ECUs) and a significant amount of memory (60.5 GBytes). The very high demand for these instances has caused that Amazon currently limits their use to only two simultaneous HI1 instances per user.

Regarding the hardware characteristics of these instances (see Table 1), the provider details the specific processor: two Intel Xeon X5570 quad-core Nehalem processors for CC1, hence 8 cores per CC1 instance, two Intel Xeon E5-2670 octa-core Sandy Bridge processors for CC2, hence 16 cores per CC2 instance, and two Intel Xeon E5620 quad-core Westmere processors for HI1, hence 8 cores per HI1 instance. Each instance will be allocated to users in a dedicated manner (a single VM per physical node), unlike the allocation mode in most other EC2 instance types (multiple VMs per physical node). These instances are interconnected via a high-speed network (10 Gigabit Ethernet), which is also among the main differential characteristics of these resources. Moreover, these instances can be launched within a placement group to obtain low latency, full bisection 10 Gbps bandwidth between them, but with the important restriction that only instances of the same type can be included in the same group.

Related with CC instances is the Cluster GPU Quadruple Extra Large Instance (cg1.4xlarge, abbreviated as CG1). Instances of this family provide exactly the same hardware capabilities than CC1 in terms of CPU power, memory capacity, I/O storage and network performance. The differential feature of CG1 instances is the provision of two GPUs for General-Purpose GPU computing (GPGPU). As the main goal of this work is the evaluation of the I/O storage subsystem, which is the same in CC1 and CG1, the study of CG1 instances has not been considered.

3.1 Storage System Overview of Amazon EC2

The virtual machines available in Amazon EC2 provide several storage solutions with different levels of abstraction, performance and access interfaces. Generally, each instance can access three types of storage: (1) the local block storage, known

Table 1 Description of the Amazon EC2 CC1, CC2 and HI1 instance types

	CC1 (cc1.4xlarge)	**CC2 (cc2.8xlarge)**	**HI1 (hi1.4xlarge)**
Release Date	July 2010	November 2011	July 2012
CPU	2 × Intel Xeon X5570 Nehalem-EP @2.93 GHz	2 × Intel Xeon E5 2670 Sandy Bridge-EP @2.60 GHz	2 × Intel Xeon E5620 Westmere-EP @2.40 GHz
ECUs	33.5	88	35
#Cores	8	16	8
Memory	23 GBytes DDR3	60.5 GBytes DDR3	60.5 GBytes DDR3
Ephemeral Storage	1.7 TBytes (2 HDD)	3.4 TBytes (4 HDD)	2 TBytes (2 SSD)
API name	cc1.4xlarge	cc2.8xlarge	hi1.4xlarge
Price (Linux)	$1.30 per hour	$2.40 per hour	$3.10 per hour
Interconnect	10 Gigabit Ethernet (Full bisection bandwidth)		
Virtualization	Xen HVM 64-bit platform		

as ephemeral disk, where user data are lost once the instances are released (non-persistent storage); (2) off-instance Elastic Block Store (EBS), which are remote volumes accessible through the network that can be attached to an EC2 instance as block storage devices, and whose content is persistent; and (3) Simple Storage Service (S3), which is a distributed object storage system, accessed through a web service that supports both SOAP and REST. We have not considered S3 in our evaluation since, unlike ephemeral and EBS devices, it lacks general file system interfaces required by scientific workloads so that the use of S3 is not transparent to the applications, and also due to the poor performance shown by previous recent works [19].

The ephemeral and EBS storage devices have different performance characteristics and usage constraints. On the one hand, a CC1 instance can only mount up to two ephemeral disks of approximately 845 GBytes each one, resulting in a total capacity of 1,690 GBytes (see Table 1), whereas a CC2 instance can mount up to four disks of the aforementioned size, 845 GBytes, which represents an overall capacity of 3,380 GBytes. The new HI1 instances, as mentioned before, provide two SSD disks of 1,024 GBytes each one as ephemeral storage, for a total of 2,048 GBytes. On the other hand, the number of EBS volumes attached to instances can be almost unlimited, and the size of a single volume can range from 1 GByte to 1 TByte.

4 Experimental Configuration and Evaluation Methodology

The I/O performance evaluation of the Amazon platform has been conducted on CC1, CC2 and HI1 instance types. This evaluation consists of a micro-benchmarking with IOzone benchmark [2] of a local file system (ext3) on ephemeral disks and EBS volumes, using a single storage device as well as multiple storage devices combined in a single software RAID 0 (data striping) array using the *mdadm* utility, as this RAID level can improve both write and read performance without losing overall capacity. The IOzone benchmark has also been used to evaluate the performance of a representative distributed file system, NFS version 3, selected as it is probably the most commonly used network file system. Additionally, it remains as the most popular choice for small and medium-scale clusters.

After characterizing NFS with the IOzone benchmark, the performance of several I/O interfaces commonly used in scientific applications has been analyzed using the IOR benchmark [33] with multiple NFS clients. Three I/O interfaces were tested: (1) POSIX, which is the IEEE Portable Operating System Interface for computing environments that defines a standard way for an application to obtain basic services from the operating system, the I/O API among them; (2) MPI-IO [35], which is a comprehensive API with many features intended specifically to provide a high performance, portable, and parallel I/O interface to MPI programs; and (3) HDF5 [4], which is a data model, library, and file format for storing and managing data that supports an unlimited variety of datatypes, and is designed for flexible and efficient I/O and for high volume and complex data. Additionally, the scalability of the parallel BT-IO kernel [38], which is implemented on top of MPI-IO, has also been analyzed at the application level both in terms of performance and cost metrics.

The pattern of I/O the operations performed by the benchmarks/applications previously selected is essentially sequential, both for write and read operations. Random accesses, widely used in some scenarios such as database environments, are rarely used in HPC applications and scientific workloads. Most parallel I/O in HPC involves large-scale data movements, such as checkpointing the state of a running application, which makes that I/O access is mainly dominated by sequential operations.

CC1 and CC2 resources have been allocated simultaneously in the same placement group in order to obtain nearby instances, thus being able to benefit from the low latency and full bisection of the 10 Gigabit Ethernet network. However, Amazon's current restriction for HI1 instances (only two HI1 instances can run simultaneously) has severely determined the evaluation. As one HI1 instance is needed to run the NFS server, the other HI1 instance can be used for the NFS clients. While this configuration is enough for the characterization of NFS with IOzone (as only one NFS client is used), for the IOR and BT-IO benchmarks it would be necessary to launch 8 HI1 instances in order to run up to 64 NFS clients. Therefore, for the evaluation of HI1 with these benchmarks, we opted for the use of one HI1 instance for the NFS server and CC1 and CC2 instances for NFS clients. This fact implies that the HI1 server and CC1/CC2 clients can not be allocated in the same placement group, because only instances of the same type can be included in it, which can cause a loss in network performance. In order to minimize it, the HI1 server was always executed in the same clients' availability zone. Thus, all the experiments were performed in the us-east-1 region (North Virginia), within the us-east-1d availability zone.

Regarding software settings, the Amazon Linux AMI 2012.03 was selected as it is a supported and maintained Linux image provided by AWS for its usage on Amazon EC2 CC instances. This AMI, which comes with kernel 3.2.18, was customized with the incorporation of the previously described benchmarks: IOzone version 3.405 and IOR version 2.10.3. In addition, the MPI implementation of the NPB suite version 3.3 was also installed for the BT-IO kernel evaluation. The metrics considered for the evaluation of the BT-IO kernel are MOPS (Millions of Operations Per Second), which measures the operations performed in the benchmark (that differ from the CPU operations issued), and its corresponding

I/O aggregated bandwidth measured in MBytes/sec. Moreover, the BT-IO Class C workload has been selected because it is the largest workload that can be executed in a single CC1 instance. The GNU C/Fortran 4.4.6 compiler has been used with -O3 flag, whereas the message-passing library selected for IOR and BT-IO evaluation is Open MPI [15], version 1.4.5. Finally, the performance results presented in this paper are the mean of the five measurements performed for each evaluated configuration. Unlike non-dedicated instance types, the dedicated (one VM per physiscal node) CC and HI1 instances present reduced variability in the performance measurements, so the standard deviation is not significant.

5 Evaluation of I/O Performance on Amazon EC2

This section presents an analysis of the performance results of the I/O subsystem on a cloud computing infrastructure, Amazon EC2 Cluster Compute platform and High I/O instances, using the representative benchmarks described in the previous section.

5.1 Local File System Performance

Figure 1 presents the maximum bandwidth obtained (measured in MBytes per second) for sequential read and write operations using the IOzone benchmark on EBS volumes and ephemeral disks (labeled as "EPH" on the graphs) on a single device formatted with the ext3 file system for CC1, CC2 and HI1 instances. The Linux buffer cache was bypassed using direct I/O (O_DIRECT flag) in order to get the real performance of the underlying storage devices.

All the configurations perform very similarly for the read operation, achieving all of them at least 100 MBytes/sec. The exception here is the SSD (ephemeral) disk on the HI1 instance, which is clearly the best performer in this case as it is able to get 900 MBytes/sec, nine times better than the rest of configurations. Regarding the write operation, the results are more insightful as EBS volumes can obtain only a 40%-50% of the performance of ephemeral disks on CC1 and CC2 instances. For writing on EBS volumes, which requires network access, the maximum bandwidth is 50.8 MBytes/sec on CC1 and 44.5 MBytes/sec on CC2, whereas for ephemeral disks the maximum bandwidth is 78.2 and 89.4 MBytes/sec on CC1 and CC2, respectively. In addition, the HI1 instance type presents similar results than CC1 and CC2 instances when using EBS volumes. Although 10 Gigabit Ethernet is available for communication among CC instances, the interconnection technology to access EBS volumes is not known. However, HI1 obtains again the best performance using the SSD disk, which obtains 562 MBytes/sec, around six times higher than the best result for the ephemeral disk on CC2. These results show that the ephemeral disks on HI1 instances provide very high performance that seems not to be affected at all by the virtualization layer.

Figure 2 presents the maximum bandwidth obtained when 2 (left graph) or 4 devices (right graph) are combined into a single software RAID 0 array (chunk size was configured at 64 KBytes). For the 2-device array configurations and the read operation, the results in CC1 and CC2 with EBS volumes and ephemeral disks are again very similar as they achieve around 200 MBytes/sec, double the

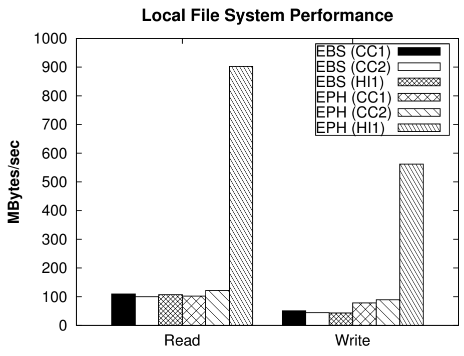

Fig. 1 Local file system (ext3) performance on Amazon EC2 CC and High I/O instances

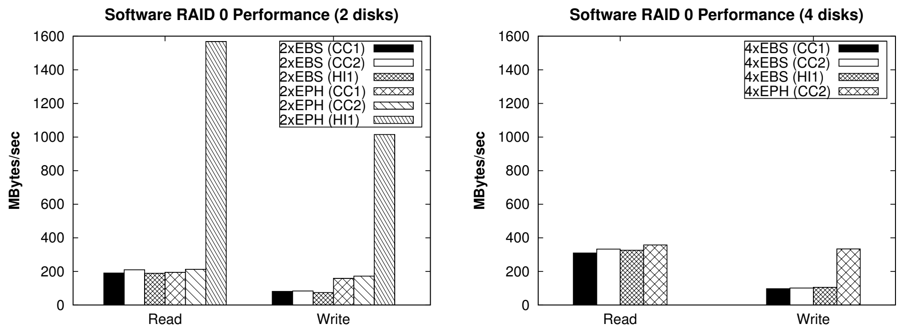

Fig. 2 Software RAID 0 performance on Amazon EC2 CC and High I/O instances

performance of a single device, which suggests that the use of software RAID and virtualization in this scenario is not harming performance. The HI1 instance type, which is obviously the best option again, gets up to 1567 MBytes/sec, 75% improvement compared to a single SSD disk. For the write operation, the performance of ephemeral disks on CC1 an CC2 instances is again significantly better than the results obtained with EBS volumes, approximately doubling its performance, whereas the SSD array is able to achieve up to 1014 MBytes/sec, which represents around 80% improvement.

Regarding 4-device array configurations (right graph), the option with ephemeral disks is only available for CC2 instances, as CC1 and HI1 are physically limited to 2 ephemeral disks by Amazon, whereas the number of EBS volumes attached to them can be almost unlimited. For the read operation, 4 EBS volumes combined are able to get a 60% improvement compared to the 2-device array (e.g., from 209 MBytes/sec to 333 MBytes/sec on CC2). However, the combination of 4 ephemeral disks on CC2 slightly outperforms EBS for reading, achieving a maximum of 358 MBytes/sec. For the write operation, although the use of two EBS volumes is able to double the performance of a single volume (83.7 vs 44.5 MBytes/sec on CC2, respectively), when 4 volumes are combined the maximum bandwidth obtained is only 101.3 MBytes/sec on CC2 and 105.3 MBytes/sec on HI1, showing poor scalability (this is the reason why the number of EBS volumes was limited to 4 in the evaluation). This poor result shows that the write performance on EBS is severely limited by the network performance. Furthermore, although the three instance types can not effectively take advantage of using 4 EBS volumes for the write operation, the combination of 4 ephemeral disks on CC2 is clearly the best performer, obtaining up to 334 MBytes/sec, showing almost a linear speedup (172 and 334 MBytes/sec for 2-device and 4-device arrays, respectively). Nevertheless, CC2 instances can not rival HI1 instances at all, as the 2-device SSD-based array on HI1 obtains more than 4 times higher read performance and up to three times more write performance than the 4-device array on CC2.

This evaluation has shown that EBS volumes suffer a significant performance penalty for write operations, and the use of software RAID can only help to partly alleviate this issue using up to 4 volumes. Therefore, the ephemeral disks, especially in RAID configuration, are the best option in a local file system scenario, as write operations are highly used in scientific applications. In fact, the SSD-based ephemeral disks of the new released HI1 instances clearly become the best choice, as they provide significantly higher performance than CC1 and CC2 ephemeral disks, both for read and write. Another advantage of using ephemeral disks is that the cost of their use is free, whereas the use of EBS volumes is charged by Amazon. However, if a particular application requires data persistence as a strong constraint, EBS volumes should be used for data safety reasons. In this scenario, a combination of both ephemeral and EBS volumes could be used.

5.2 Distributed File System Performance

Figures 3 and 4 present the results of the read and write performance of a distributed file system, NFS, with a base configuration of one instance running the NFS server and one client instance connecting to the server through the 10 Gigabit Ethernet network. The micro-benchmark selected is the IOzone benchmark using both EBS volumes and ephemeral disks as storage devices, and using different file sizes for CC1, CC2 and HI1 instances. In these experiments, as only two instances are needed (one for the server and one for the client), two HI1 instances (the maximum that can be launched) allocated in the same placement group have been used, as for CC1 and CC2. For clarity purposes, the figures only present experimental results from RAID configurations as they provide better performance. In these experiments the NFS server buffer cache (or page cache) has not been bypassed in order to reflect the performance results of a typical NFS configuration, which generally takes advantage of this mechanism to achieve higher performance.

Two NFS server configurations for the write operation have been used: (1) asynchronous (*async*) mode, which can provide high performance as it supports NFS calls to return the control to the client before the data has been flushed to disk; and (2) synchronous (*sync*) mode, where a block has to be actually written to disk before returning the control to the client, providing higher data safety in terms of data persistence when the NFS server crashes. In addition, the Amazon AMI for CC instances sets the Maximum Transmission Unit (MTU) of the network to 1,500 bytes by default. In order to assess the impact of the use of Jumbo frames (MTUs

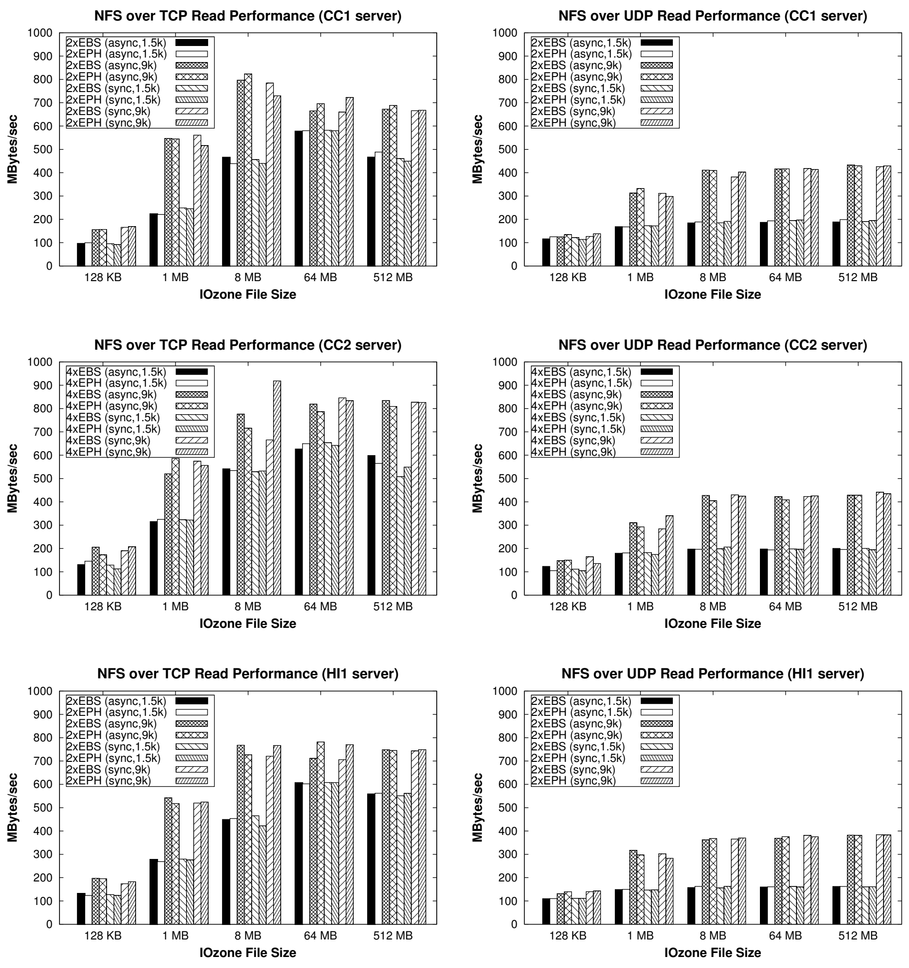

Fig. 3 NFS read performance through 10 Gigabit Ethernet on Amazon EC2

higher than the default 1,500 bytes) the tests have been repeated with the MTU configured at 9,000 bytes (maximum MTU value supported) both for server and client. Moreover, TCP and UDP transport protocols have been tested in order to characterize the impact of the selected protocol on the overall performance, which is highly important in virtualized environments where the network plays a key role. Both transport protocols have been configured with the maximum block size allowed for each one in this kernel version (1 MByte and 32 KBytes for TCP and UDP, respectively). These are the parameters that have generally shown a significant impact on performance. Finally, *noatime* and *nodiratime* mount options were enabled in the client as they can provide a small performance gain.

Figure 3 shows performance results on CC1, CC2 and HI1 instances (from top to bottom) which compare TCP and UDP protocols for the read operation under the different settings considered. TCP results (left graphs) significantly outper-

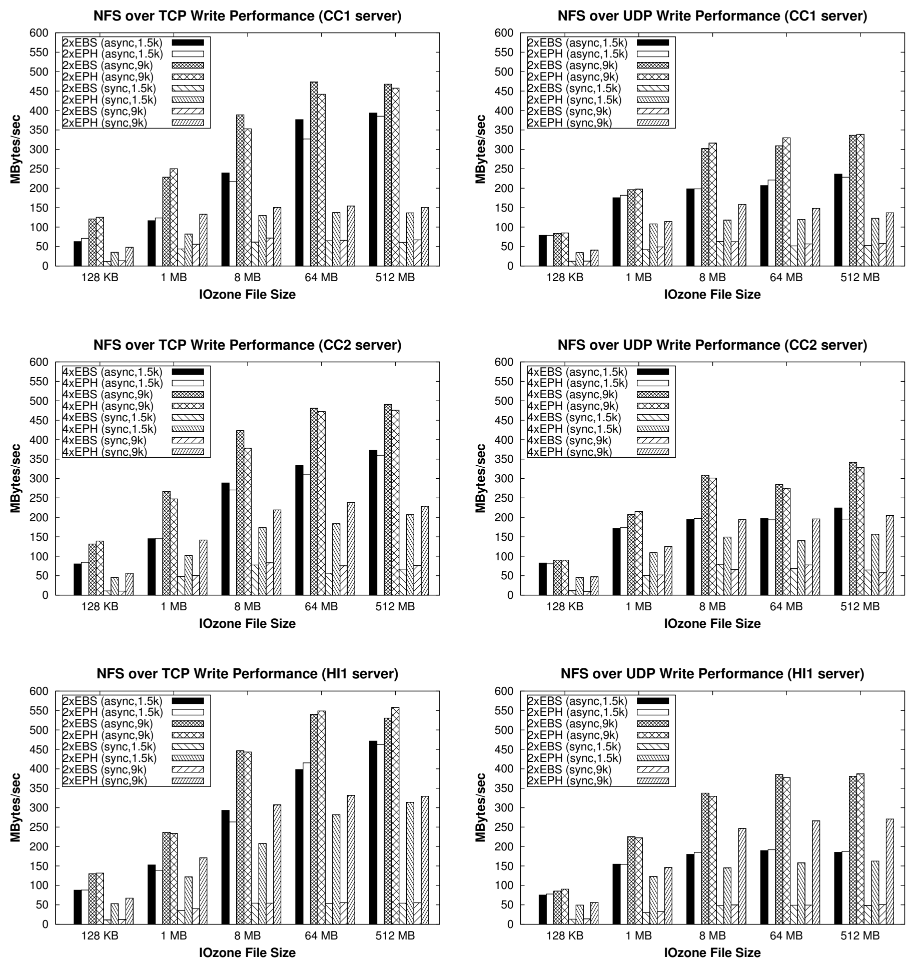

Fig. 4 NFS write performance through 10 Gigabit Ethernet on Amazon EC2

form UDP (right graphs) for all file sizes and configurations, especially from 8 MBytes on. This fact is due to the higher block size allowed in the TCP protocol, which benefits TCP especially when file sizes larger than 1 MByte are used. The performance comparison between EBS volumes and ephemeral disks for both protocols results in a tie, since data is actually read from the NFS server buffer cache that hides the underlying storage performance. In addition, the server *async* configuration is not able to achieve better performance as only write operations can take full advantage of this feature. However, the MTU parameter presents a huge impact on performance for both protocols; in fact in some cases the 9,000 byte MTU value allows for up to 165% of improvement over using the default MTU value of 1,500 bytes (e.g., see the 1 MByte file size for TCP protocol on CC1). The comparative analysis among different instance types shows that they achieve generally similar read performance, once again due to the operation of the NFS server cache.

Figure 4 compares TCP and UDP protocols for the write operation. In this case, these results have revealed some important facts: (1) TCP performance is again generally higher than UDP performance, except for the smallest file sizes, 128 KBytes and 1 MByte, where UDP slightly outperforms TCP when MTU is 1,500 bytes; (2) the *async* server configuration is able to provide significantly higher performance (up to 4 times better) than *sync* configuration for both protocols and all file sizes; (3) increasing the MTU value to 9,000 bytes provides better performance results for all configurations, especially for the *async* mode; (4) ephemeral disks show significantly better results than EBS volumes in the *sync* mode, confirming some of the conclusions derived from the the local file system benchmarking. However, the *async* server mode allows to reduce the performance penalties of using EBS volumes enabling the overlapping of I/O, as control is returned to the client when data is written in the server buffer cache (the actual writing to disk is done asynchronously), which results in similar performance for EBS volumes and ephemeral disks in this scenario; (5) the *async* mode shows very similar results on CC1 and CC2 (for both protocols), whereas HI1 seems to perform slightly better than CC instances especially when using TCP and large file sizes; and (6) the *sync* mode allows to analyze more straightforwardly the underlying storage system performance. Thus, CC1 achieves up to 154 MBytes/sec on TCP with ephemeral disks, showing that performance in this case is being limited by the 2-device array. CC2, relying on an array of 4 ephemeral disks, outperforms CC1 with 238 MBytes/sec, whereas HI1 obtains up to 332 MBytes/sec thanks to the use of SSD disks. However, these results on CC2 and HI1 instances with *sync* mode reveal that the network becomes the main performance bottleneck, reducing significantly the maximum underlying disk performance obtained in the Section 5.1 (334 and 1014 MBytes/sec for CC2 and HI1, respectively).

5.2.1 Multi-client NFS Performance Using Different I/O Interfaces

Figure 5 shows the aggregated read (left graphs) and write (right graphs) bandwidths, measured in MBytes/sec, obtained with the parallel I/O IOR benchmark in a configuration with one NFS server exporting multiple devices, either EBS volumes or ephemeral disks, and with multiple NFS clients which access the server through a 10 Gigabit Ethernet network. The experimental configuration of this micro-benchmarking includes an optimal combination of values of NFS parameters, in order to provide the highest performance, that is to say the *async* mode for the NFS server as well as the use of the TCP protocol for NFS clients. The MTU value has also been configured at 9,000 bytes on all the machines involved, replacing the default value.

In these experiments, each of the client instances runs 8 (on CC1) or 16 (on CC2) parallel processes, reading and writing a single shared file collectively. For CC1 and CC2, both server and clients are instances of the same type. However, as mentioned in Section 4, the current restriction in the use of the HI1 instance type (only two instances can run simultaneously) limits the configuration of the HI1 testbed to the use of one HI1 instance for the NFS server and either CC1 or CC2 instance types for the NFS clients, which also implies that server and clients can not be allocated in the same placement group. EBS results on this HI1 testbed

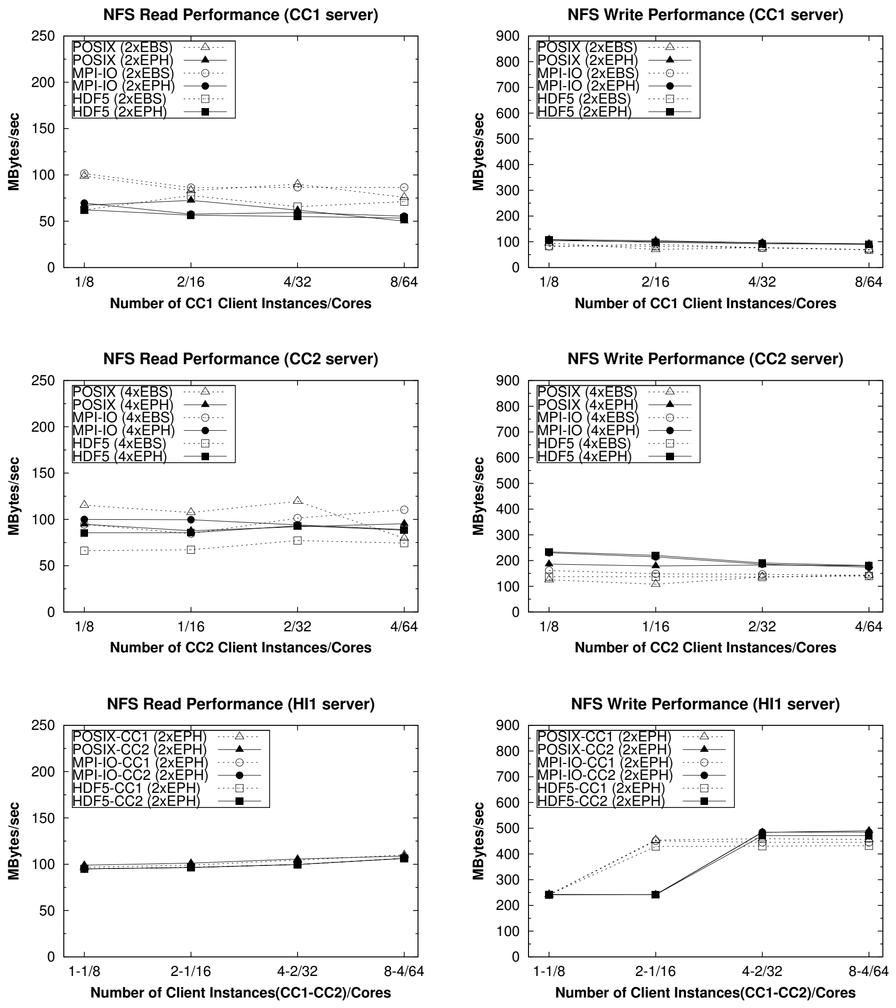

Fig. 5 NFS performance using multiple clients on Amazon EC2 CC and High I/O instances

have been omitted for clarity purposes, since they are very similar to those of CC1 (for a 2-device array) and CC2 (for a 4-device array).

In order to obtain the underlying I/O throughput, the aggregated file size has been configured for each test to ensure that the NFS server memory is exhausted. This means that the aggregated file size under consideration is significantly larger than the available memory in the server, 23 and 60.5 GBytes for CC1 and CC2/HI1, respectively, reducing the impact of the usage of the NFS server buffer cache on the results. Finally, it has been set a high value (16 MBytes) for the transfer size parameter of IOR, which represents the amount of data transferred per process between memory and file for each I/O function call, in order to achieve the maximum possible aggregated bandwidth.

Performance results on CC1, CC2 and HI1 instances for the read operation are presented in the left graphs. As can be observed, CC2 obtains better results than CC1 for both storage devices and all interfaces (except for HDF5 on EBS which performs very similarly) due to the use of a 4-device array. On CC1 instances EBS volumes outperform ephemeral disks for the three evaluated interfaces, whereas this fact only occurs for MPI-IO on CC2. This result confirms that ephemeral disks scale better than EBS volumes when the number of devices combined in the RAID array increases, as seen in the local file system benchmarking results. The comparison between the interfaces shows that their performance is quite similar, although HDF5 results are slightly worse than the others for the read operation. On average, MPI-IO is the best performer for CC instances. Regarding the HI1 read graph, it shows almost similar results for the three interfaces evaluated on ephemeral disks for both CC1 and CC2 clients, and also very similar to CC2 results. This fact confirms that the network is the main performance bottleneck and thus the availability of a high performance storage device does not improve performance significantly.

The right graphs of Figure 5 present the results for the write operation, where CC2 clearly outperforms CC1 instance type again, doubling the performance in some cases. Moreover, ephemeral disks are able to obtain better performance than EBS, even when the number of processes increases. Here, the additional network access incurred using EBS volumes seems to be the responsible for this performance penalty. Regarding HI1 instance type, the results show again that the network is clearly limiting its overall performance, achieving up to 456 and 490 MBytes/sec with 64 clients of CC1 and CC2 instance types, respectively. In addition, the performance difference between the I/O interfaces is almost null, and the difference between CC1 and CC2 clients is also negligible for 8, 32 and 64 clients. However, the use of a single CC2 instance to run 16 client processes (as it has 16 cores) obtains significantly lower (around half) performance than using two CC1 instances, where each one runs 8 client processes. This fact suggests that, once again, the performance bottleneck is in the network access, as the CC2 instance client has twice processes (16) accessing simultaneously the network card, thus dividing the available bandwidth per process and showing a poor ratio between network and CPU performance. For the remaining scenarios (8, 32 and 64 clients) the network link between the server (HI1) and the clients (CC1/CC2) remains as the limiting factor for the overall performance.

These results have revealed an important fact: the poor virtualized network performance clearly limits the new HI1 instances with SSD disks, especially for the read operation. Nevertheless, HI1 instances can provide better performance (up to twice higher) than CC instances for the write operation when the NFS server is configured in the *async* mode. Additionally, these results have confirmed the higher performance of the ephemeral disks for the write operation compared to EBS volumes, especially when using the CC2 and HI1 resources. However, for the read operation EBS volumes can achieve similar or even better performance than ephemeral disks. Therefore, the choice between storage devices, EBS or ephemeral, will depend on the I/O characteristics and requirements of each particular application.

5.2.2 The Effect of Caching and Transfer Size on NFS Performance

In Section 5.2.1, the NFS server buffer cache was exhausted writing a shared file size which was significantly larger than the server memory in order to ensure that the performance of the underlying storage subsystem was actually being measured. This section presents the analysis of the effect of caching by writing a shared file which size is less than the server memory, and under different transfer sizes (from 16 KBytes to 16 MBytes), using MPI-IO as a representative I/O interface. The results for CC1, CC2 and HI1 instances using 64 clients are shown in Figure 6, under the same NFS configuration than in the previous subsection but only including ephemeral disks that they provide the best write performance.

The left graph of Figure 6 shows the performance results of writing a large file, twice larger than the available memory on the NFS server (64 GBytes for CC1 and 128 GBytes for CC2 and HI1). The results using a transfer size of 16 MBytes are the same as those shown in the previous subsection for MPI-IO (see Figure 5). The right graph shows the performance of writing a file size that fits into the available memory of the server (16-GByte file size). Performance results have been obtained for 4 different configurations: (1) a CC1 server and 8 CC1 clients (CC1-CC1); (2) a CC2 server and 4 CC2 clients (CC2-CC2); (3) an HI1 server and 8 CC1 clients (HI1-CC1); and (4) an HI1 server and 4 CC2 clients (HI1-CC2). The limitation in the number of available HI1 instances (right now up to 2 per user) has prevented the evaluation of a scenario with both server and clients in HI1 instance. However, in this hypothetical configuration (HI1-HI1) the clients would benefit from being located in the same placement group, but they will not take advantage of the locally attached SSD disks and will suffer from the limited computational power of the HI1 systems (35 ECUs, similar to CC1 instances, but far from the 88 ECUs of CC2 instances).

The first conclusion that can be derived from this analysis is that the use of the largest transfer size (16 MBytes in this scenario) is key to achieve high parallel I/O performance, mainly HI1 instances. The results in the left graph clearly show that the SSD disks on HI1 provide significantly better performance from 64 KBytes on. Once the server cache is exhausted, performance is determined by the I/O subsystem, although for HI1 performance is ultimately determined by the network, as seen in Section 5.2.1.

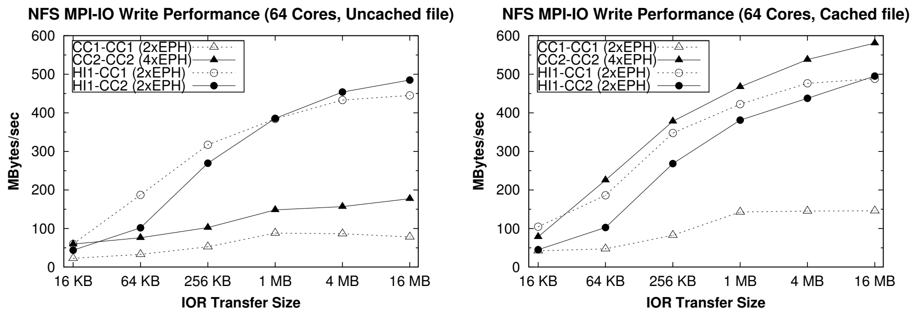

Fig. 6 NFS performance depending on caching and transfer size

The right graph shows significant performance increases for the CC2-CC2 configuration when the file is cached, whereas for the rest of configurations the results are only slightly better. The performance of the CC1-CC1 configuration is clearly limited by the poor performance of the underlying I/O subsystem, based on a 2-device array of ephemeral disks that, according to our previous local file system benchmarking only provides up to 160 MBytes/sec (see Figure 2 in Section 5.1). Regarding the HI1 server-based configurations (HI1-CC1 and HI1-CC2), whose 2-device array is able to provide above 1000 MBytes/sec, the network severely limits performance. This is mainly to the fact that the server is not in the same placement group of CC1/CC2 clients, which increases latency and reduces the full bandwidth available. Moreover, the CC2-CC2 configuration is now the best performer, even slightly better than the HI1 server-based configurations, due to a combination of two facts: (1) unlike the HI1 configurations, the allocation of CC2 server and clients in the same placement group enables to exploit the full network bandwidth; and (2) a CC2 instance provides higher memory write performance (up to 20% more) than an HI1 instance, according to the STREAM benchmark [22], which clearly benefits the operation of the NFS buffer cache in a CC2 server.

5.3 I/O-intensive Parallel Application Performance

The performance of a representative I/O-intensive parallel application, the BT-IO kernel from the NPB suite, has been analyzed. This code is the I/O-enabled version of the NPB BT benchmark, which solves Navier-Stokes equations in three spatial dimensions. As mentioned in Section 4, the NPB Class C workload has been selected, whereas the I/O size is the Full subtype. With these settings, all processes append data to a single file through 40 collective MPI-IO write operations, generating a total of 6.8 GBytes of output data, which are also read at the end of the execution. It has been used the default I/O frequency, which consists of appending data to the shared output file every 5 computation time steps. Finally, the BT-IO evaluation has been performed using the same NFS configuration as in Section 5.2, as it maximizes the NFS performance.

Figure 7 presents BT-IO performance using up to 64 clients. The performance metrics reported are the aggregated bandwidth measured in MBytes/sec (left graph) and MOPS (right graph). BT-IO requires that the number of client processes must be square numbers.

The aggregated bandwidth results confirm that ephemeral disks can provide better performance than EBS volumes. Thus, the CC1-CC1 configuration with ephemeral devices achieves up to 271 MBytes/sec (for 64 clients) whereas EBS volumes obtain up to 254 MBytes/sec. The numbers for the CC2-CC2 configuration are 327 and 302 MBytes/sec, respectively. Regarding the HI1 server-based testbeds, only ephemeral (SSD) devices have been considered, showing the best result for CC2 clients (e.g., 338 MBytes/sec for 64 clients). Here, the need of double the number of CC1 instances than CC2 ones (8 vs 4 when considering 64 clients) represents an important performance bottleneck for this code, as BT-IO is also a computation/communication-intensive code. Thus, the higher number of network communications required by CC1 are significantly affected by the poor virtualized network performance. This fact causes that the CC2-CC2 configuration using ephemeral devices outperforms HI1-CC1 for 36 and 64 clients.

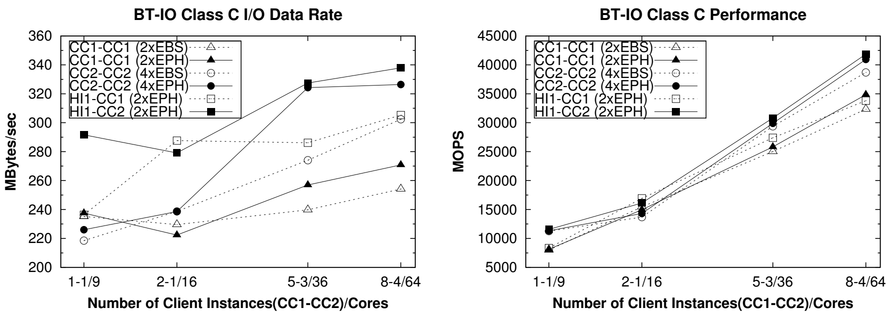

Fig. 7 BT-IO performance on Amazon EC2 CC and High I/O instances

These assessments are confirmed by the overall performance of the application in terms of MOPS (right graph). Thus, HI1-CC1 achieves similar results to CC1-CC1 (less than 35,000 MOPS), while using CC2 clients (either for CC2-CC2 or HI1-CC2) the measured performance is up to 20% higher (e.g., 42,000 MOPS for HI1-CC2). Here, the differences in I/O performance are not translated into equivalent differences in MOPS, because I/O represents around 25-35% of the total execution time of the application on these scenarios. The remaining execution time is spent in computation and MPI communications, increasing the communication overhead with the number of processes, which explains the impact of MPI communications on the overall performance as well as on I/O performance since disk writes are performed collectively.

Taking into account the different instance types, the use of a 4-device array allows CC2 to achieve higher performance than CC1 both for ephemeral and EBS devices, especially when considering the aggregated bandwidth, where CC2 servers obtain approximately 20% more bandwidth than CC1 servers. As mentioned before, the HI1-CC1 configuration obtains poor results, whereas HI1-CC2 slightly outperforms the CC2-CC2 testbed on 64 clients (338 vs 327 MBytes/sec, respectively). The fact that all data are cached in the NFS server (in this scenario, 6.8 GBytes are written to disk) together with the poor performance of the virtualized network between the NFS server and its clients prevented the HI1 server-based configurations (in particular, HI1-CC2) to obtain better performance.

5.3.1 Cost Analysis of I/O-intensive Parallel Codes

Amazon EC2 offers different purchasing options: (1) on-demand instances, which allow to access immediately computation power by paying a fixed hourly rate; (2) spot instances from the spot market, which allow customers to bid on unused Amazon EC2 capacity and run those instances for as long as their bid exceeds the current spot price (which changes periodically based on supply and demand); and (3) reserved instances for one- or three-year terms, which allow to receive a significant discount on the hourly charge. There are three reserved instance types: light, medium and heavy, that enable to balance the amount payed upfront with the effective hourly price. Table 2 presents all the prices considered in the analysis.

Table 2 EC2 pricing for CC1, CC2 and HI1 (Linux/UNIX) instance types (us-east-1 region)

Instance Type	On-demand	Spot Price	Reserved (3-year term)
CC1	$1.30	$0.818	$0.537
CC2	$2.40	$0.948	$0.653
HI1	$3.10	Not available	$0.899

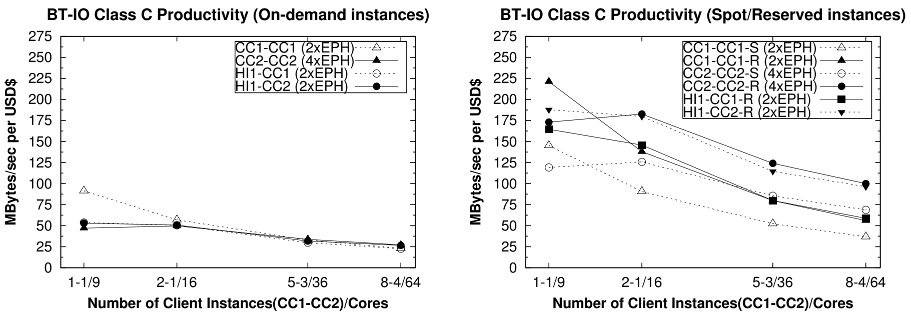

Fig. 8 BT-IO productivity using on-demand, spot and reserved instances

In order to ease the analysis of the cost of Amazon EC2 resources, Figure 8 presents the productivity of the previously evaluated BT-IO application in terms of aggregated bandwidth per USD$. Only results for ephemeral disks are shown for clarity purposes as they provide better performance. Different purchasing options are compared: (1) using the price of on-demand instances (left graph); (2) using the average spot price in the July-September 2012 period (right graph, labeled as "S"), only for CC1 and CC2 instances as currently HI1 instances are not offered in the spot market; and (3) using the price calculated with heavy utilization reserved instances for a three-year term (right graph, labeled as "R"), which usually represents the lowest price that can be obtained for a particular instance type.

On the one hand, the results using on-demand instances (left graph) show negligible differences between the different options except for the CC1-CC1 configuration with 9 clients, which is the most cost-effective option. On the other hand, the results using spot and reserved instances (right graph) present more differences among the evaluated configurations. First of all, the use of spot instances can provide significant cost improvements, up to 3 times higher performance/cost ratio than using on-demand instances (e.g., for CC2-CC2 with 64 clients), although here the main drawback of spot instances is that they can be shut down at any moment (when the spot price moves higher than the customer's maximum price). Regarding reserved instances, they allow for up to 25% higher performance/cost ratio on average than using spot instances, being CC2-CC2 the most cost-effective option, slighly better than HI1-CC2. Furthermore, reserved instances will be always active for the availability zone specified at purchase time. Thus, reserved instances are the best choice in terms of performance/cost ratio.

6 Conclusions

Cloud computing is a model that enables on-demand and self-service access to a pool of highly scalable, abstracted infrastructure, platform and/or services, which are billed by consumption. This paradigm is currently being explored by the scientific community to assess its suitability for HPC applications. Among current cloud providers, Amazon WS is the leading commercial public cloud infrastructure provider.

This work has presented a comprehensive evaluation of the I/O storage subsystem on the Amazon EC2 Cluster Compute platform, a family of instance types which are intended to be well suited for HPC applications. Moreover, this work has included the evaluation of the new High I/O (HI1) instance type recently released by Amazon (July 2012), which provides SSD disks as ephemeral storage, thus highly oriented to be used in scalable cloud storage I/O systems. The performance evaluation was carried out at different layers and using several representative micro-benchmarks. Thus, the cloud low-level storage devices available in these instances (ephemeral disks and EBS volumes) have been evaluated both on local and distributed file systems, as well as the performance of several I/O interfaces commonly used in scientific applications (POSIX, MPI-IO and HDF5). Moreover, the scalability of an I/O-intensive code, the BT-IO application from the NPB suite, has also been analyzed at the application level, including an analysis in terms of cost.

Performance results have shown that the available cloud storage devices present significant performance differences. Thus, this paper has revealed that the use of ephemeral disks can provide more performance than EBS volumes for the write operation, especially when software RAID is used, thanks to the avoidance of additional network accesses to EBS, outside of the placement group, as EBS performance is deeply influenced by the network overhead and variability. In addition, this paper has characterized NFS performance on Amazon EC2, showing the impact of the main NFS configuration parameters on a virtualized cloud environment. Moreover, the analysis of the parallel I/O performance on Amazon EC2 has revealed that HI1 instances can provide significantly better write performance than any other instance type when writing very large files with large transfer sizes, although the overall performance is ultimately limited by the poor network throughput. Finally, the analysis of the performance/cost ratio of the BT-IO application has shown that, although the use of the HI1 instance type provides slightly better raw performance in terms of aggregated bandwidth, it may not be the best choice when taking into account the incurred costs.

Acknowledgements This work has been funded by the Ministry of Science and Innovation of Spain [Project TIN2010-16735 and an FPU Grant AP2010-4348], by the Galician Government [Program for the Consolidation of Competitive Research Groups, ref. 2010/6] and by an Amazon Web Services (AWS) LLC research grant.

References

1. Amazon Web Services in Top 500 list. `http://www.top500.org/system/177457`. Last visited: November 2012
2. IOzone Filesystem Benchmark. `http://www.iozone.org/`. Last visited: November 2012

3. MPI: A Message Passing Interface Standard. `http://www.mcs.anl.gov/research/projects/mpi/`. Last visited: November 2012
4. The HDF Group. `http://www.hdfgroup.org/HDF5/`. Last visited: November 2012
5. Abe, Y., Gibson, G.: pWalrus: Towards better integration of parallel file systems into cloud storage. In: Workshop on Interfaces and Abstractions for Scientific Data Storage (IASDS'10), pp. 1–7. Heraklion, Crete, Greece (2010)
6. Amazon Web Services LLC: Amazon Elastic Block Store (EBS). `http://aws.amazon.com/ebs/`. Last visited: November 2012
7. Amazon Web Services LLC: Amazon Elastic Compute Cloud (Amazon EC2). `http://aws.amazon.com/ec2`. Last visited: November 2012
8. Amazon Web Services LLC: Amazon Simple Storage Service (Amazon S3). `http://aws.amazon.com/s3/`. Last visited: November 2012
9. Amazon Web Services LLC: High Performance Computing Using Amazon EC2. `http://aws.amazon.com/ec2/hpc-applications/`. Last visited: November 2012
10. Carns, P., Ligon III, W., Ross, R., Thakur, R.: PVFS: A parallel virtual file system for linux clusters. In: Proc. 4th Annual Linux Showcase & Conference, pp. 317–328. Atlanta, GA, USA (2000)
11. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Communications of the ACM **51**(1), 107–113 (2008)
12. Deelman, E., Singh, G., Livny, M., Berriman, B., Good, J.: The cost of doing science on the cloud: The montage example. In: Proc. 20th ACM/IEEE Supercomputing Conference (SC'08), pp. 50:1–50:12. Austin, TX, USA (2008)
13. Evangelinos, C., Hill, C.N.: Cloud computing for parallel scientific HPC applications: Feasibility of running coupled atmosphere-ocean climate models on Amazon's EC2. In: Proc. 1st Workshop on Cloud Computing and Its Applications (CCA'08), pp. 1–6. Chicago, IL, USA (2008)
14. Expósito, R.R., Taboada, G.L., Ramos, S., Touriño, J., Doallo, R.: Performance analysis of HPC applications in the cloud. Future Generation Computer Systems **29**(1), 218–229 (2013)
15. Gabriel, E., et al.: Open MPI: Goals, concept, and design of a next generation MPI implementation. In: Proc. 11th European PVM/MPI Users' Group Meeting (EuroPVM/MPI'04), pp. 97–104. Budapest, Hungary (2004)
16. Ghoshal, D., Canon, R.S., Ramakrishnan, L.: I/O performance of virtualized cloud environments. In: Proc. 2nd International Workshop on Data Intensive Computing in the Clouds (DataCloud-SC'11), pp. 71–80. Seattle, WA, USA (2011)
17. Gunarathne, T., Wu, T.L., Qiu, J., Fox, G.: MapReduce in the Clouds for Science. In: Proc. 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom'10), pp. 565–572. Indianapolis, IN, USA (2010)
18. Huang, W., Liu, J., Abali, B., Panda, D.K.: A case for high performance computing with virtual machines. In: Proc. 20th ACM International Conference on Supercomputing (ICS'06), pp. 125–134. Cairns, Australia (2006)
19. Juve, G., Deelman, E., Berriman, G.B., Berman, B.P., Maechling, P.: An evaluation of the cost and performance of scientific workflows on Amazon EC2. Journal of Grid Computing **10**(1), 5–21 (2012)
20. Liu, M., Zhai, J., Zhai, Y., Ma, X., Chen, W.: One optimized I/O configuration per HPC application: Leveraging the configurability of cloud. In: Proc. 2nd ACM SIGOPS Asia-Pacific Workshop on Systems (APSys'11), pp. 1–5. Shanghai, China (2011)
21. Mauch, V., Kunze, M., Hillenbrand, M.: High performance cloud computing. Future Generation Computer Systems (In press, http://dx.doi.org/10.1016/j.future.2012.03.011)
22. McCalpin, J.D.: Memory bandwidth and machine balance in current high performance computers. IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter pp. 19–25 (1995)
23. Napper, J., Bientinesi, P.: Can cloud computing reach the TOP500? In: Proc. Combined Workshops on UnConventional High Performance Computing Workshop Plus Memory Access Workshop (UCHPC-MAW'09), pp. 17–20. Ischia, Italy (2009)
24. NASA: NAS Parallel Benchmarks. `http://www.nas.nasa.gov/publications/npb.html`. Last visited: November 2012
25. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: The Eucalyptus open-source cloud-computing system. In: Proc. 9th IEEE International Symposium on Cluster Computing and the Grid (CCGRID'09), pp. 124–131. Shanghai, China (2009)

26. de Oliveira, D., Ocaña, K.A.C.S., Baião, F.A., Mattoso, M.: A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds. Journal of Grid Computing **10**(3), 521–552 (2012)
27. Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T., Epema, D.: A performance analysis of EC2 cloud computing services for scientific computing. In: Proc. 1st International Conference on Cloud Computing (CLOUDCOMP'09), pp. 115–131. Munich, Germany (2009)
28. Palankar, M.R., Iamnitchi, A., Ripeanu, M., Garfinkel, S.: Amazon S3 for science grids: A viable solution? In: Proc. 1st International Workshop on Data-aware Distributed Computing (DADC'08), pp. 55–64. Boston, MA, USA (2008)
29. Ramakrishnan, L., Canon, R.S., Muriki, K., Sakrejda, I., Wright, N.J.: Evaluating Interconnect and Virtualization Performance for high performance computing. SIGMETRICS Performance Evaluation Review **40**(2), 55–60 (2012)
30. Regola, N., Ducom, J.C.: Recommendations for virtualization technologies in high performance computing. In: Proc. 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom'10), pp. 409–416. Indianapolis, IN, USA (2010)
31. Rodero, I., Viswanathan, H., Lee, E.K., Gamell, M., Pompili, D., Parashar, M.: Energy-efficient thermal-aware autonomic management of virtualized HPC cloud infrastructure. Journal of Grid Computing **10**(3), 447–473 (2012)
32. Shafer, J.: I/O virtualization bottlenecks in cloud computing today. In: Proc. 2nd Workshop on I/O Virtualization (WIOV'10), p. 5 (7 pages). Pittsburgh, PA, USA (2010)
33. Shan, H., Antypas, K., Shalf, J.: Characterizing and predicting the I/O performance of HPC applications using a parameterized synthetic benchmark. In: Proc. 20th ACM/IEEE Supercomputing Conference (SC'08), pp. 42:1–42:12. Austin, TX, USA (2008)
34. Sun, C., Nishimura, H., James, S., Song, K., Muriki, K., Qin, Y.: HPC cloud applied to lattice optimization. In: Proc. 2nd International Particle Accelerator Conference (IPAC'11), pp. 1767–1769. San Sebastian, Spain (2011)
35. Thakur, R., Gropp, W., Lusk, E.: On implementing MPI-IO portably and with high performance. In: Proc. 6th Workshop on I/O in Parallel and Distributed Systems (IOPADS '99), pp. 23–32. Atlanta, GA, USA (1999)
36. Vecchiola, C., Pandey, S., Buyya, R.: High-performance cloud computing: A view of scientific applications. In: Proc. 10th International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN'09), pp. 4–16. Kaoshiung, Taiwan (2009)
37. Walker, E.: Benchmarking Amazon EC2 for high-performance scientific computing. ;login: The usenix journal **33**(5), 18–23 (2008)
38. Wong, P., van der Wijngaart, R.: NAS parallel benchmarks I/O version 2.4. Tech. Rep. NAS-03-002, NASA Ames Research Center (2003)
39. Yang, H., Luan, Z., Li, W., Qian, D.: MapReduce workload modeling with statistical approach. Journal of Grid Computing **10**(2), 279–310 (2012)
40. Youseff, L., Wolski, R., Gorda, B., Krintz, C.: Paravirtualization for HPC systems. In: Proc. International Workshop on XEN in HPC Cluster and Grid Computing Environments (XHPC'06), pp. 474–486. Sorrento, Italy (2006)
41. Yu, W., Vetter, J.S.: Xen-based HPC: A parallel I/O perspective. In: Proc. 8th IEEE International Symposium on Cluster Computing and the Grid (CCGRID'08), pp. 154–161. Lyon, France (2008)
42. Zhai, Y., Liu, M., Zhai, J., Ma, X., Chen, W.: Cloud versus in-house cluster: Evaluating Amazon cluster compute instances for running MPI applications. In: Proc. 23rd ACM/IEEE Supercomputing Conference (SC'11, State of the Practice Reports), pp. 11:1–11:10. Seattle, WA, USA (2011)
43. Zhang, Y., Gao, Q., Gao, L., Wang, C.: iMapReduce: A distributed computing framework for iterative computation. Journal of Grid Computing **10**(1), 47–68 (2012)